Published in final edited form as:
IEEE Trans Knowl Data Eng. 2023 April ; 35(4): 4033–4046. doi:10.1109/tkde.2022.3142773.

M²: Mixed Models With Preferences, Popularities and Transitions for Next-Basket Recommendation

Bo Peng, MS,
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA.

Zhiyun Ren, PhD,
Department of Biomedical Informatics, The Ohio State University, Columbus, OH 43210 USA.

Srinivasan Parthasarathy, PhD [Member, IEEE],
Department of Biomedical Informatics, Department of Computer Science and Engineering, and Translational Data Analytics Institute, The Ohio State University, Columbus, OH43210 USA.

Xia Ning, PhD [Member, IEEE]
Department of Biomedical Informatics, Department of Computer Science and Engineering, and Translational Data Analytics Institute, The Ohio State University, Columbus, OH43210 USA.

Abstract

Next-basket recommendation considers the problem of recommending a set of items into the next basket that users will purchase as a whole. In this paper, we develop a novel mixed model with preferences, popularities and transitions (`M`^{2}) for the next-basket recommendation. This method models three important factors in next-basket generation process: 1) users' general preferences, 2) items' global popularities and 3) transition patterns among items. Unlike existing recurrent neural network-based approaches, `M`^{2} does not use the complicated networks to model the transitions among items, or generate embeddings for users. Instead, it has a simple encoder-decoder based approach (`ed-Trans`) to better model the transition patterns among items. We compared `M`^{2} with different combinations of the factors with 5 state-of-the-art next-basket recommendation methods on 4 public benchmark datasets in recommending the first, second and third next basket. Our experimental results demonstrate that `M`^{2} significantly outperforms the state-of-the-art methods on all the datasets in all the tasks, with an improvement of up to 22.1%. In addition, our ablation study demonstrates that the `ed-Trans` is more effective than recurrent neural networks in terms of the recommendation performance. We also have a thorough discussion on various experimental protocols and evaluation metrics for next-basket recommendation evaluation.

Corresponding author: Xia Ning. ning.104@osu.edu.

Index Terms—

Recommender systems; next-basket recommendation; encoder-decoder architecture; mixed models

1 Introduction

Next-basket recommendation [1], [2], [3], [4], [5] considers the problem of recommending a set of items into the next basket that users will purchase as a whole, based on the baskets of items that users have purchased. It is different from the conventional top-N recommendation problem in recommender systems, in which users will purchase a single item at each time. Next-basket recommendation has been drawing increasing attention from research community due to its wide applications in the grocery industry [1], [3], fashion industry [6] and tourism industry [7], etc. With the prosperity of deep learning, many deep models, particularly based on recurrent neural networks (RNNs) [1], [3], [4], [5] have been developed for next-basket recommendation purposes, and have demonstrated superior performance [1], [3]. These methods, especially these RNN-based methods, often focus on modeling the transitions between different baskets, but are not always effective to model various important factors that may determine next baskets. For example, the transition among individual items in different baskets is an important factor, as given the individual items in the previous baskets, the probability of being interacted/purchased in the next basket is not equal for all the items. Users' general preference is another important factor as different users generally will have different preferences on items. Recently developed RNN-based methods [1], [3], [4] typically explicitly model the transitions among baskets, while implicitly model the transitions among individual items. For example, these methods use mean pooling or weighted sum to aggregate the items in a same basket, and then use the recurrent units to model the transitions among baskets. However, during such aggregation, the information of individual items could be smoothed out so that these methods could not accurately model the transitions among individual items. In addition, due to the recurrent nature of RNNs, it is challenging to train these RNN-based methods efficiently in parallel. Another limitation with existing methods is that, existing methods [2] usually model users' general (i.e., long-term) preferences using the embeddings of users. However, due to the notoriously sparse nature of data in recommendation problems, these learned embeddings may not be able to accurately capture users' preferences. To mitigate the limitations in the existing basket recommendation methods, in this paper, we develop a set of novel mixed models, denoted as $\mathtt{M}^2$, for the next-basket recommendation problem.

$\mathtt{M}^2$ models three important factors in order to generate next-basket recommendations for each user. The first factor is users' general preferences, which will measure long-term preferences of users that tend to remain consistent across multiple baskets during a certain period of time. The second factor is items' global popularities, which will measure the overall popularities of items among all the users. The third factor is the transition patterns among items across baskets, which will capture the transition patterns on items over different baskets. These three factors will be combined together using weights that will be determined by these factors, and thus recommend items into the next basket. With different combinations of factors, M^2 has three variants M^2-p^2, M^2-gp^2 and M^2-gp^{2}t. M^2-p^2 recommends items using users' general preferences and items' global popularities. In M^2-p^2, these two factors are combined using a global weight. M^2-gp^2 is similar to M^2-p^2 except that instead of using a global weight, M^2-gp^2 learns personalized weights to combine the two factors. M^2-gp^{2}t uses all the three factors for more accurate recommendations. The details of these three variants will be presented in Section 4. In particular, different from existing methods, M^2 explicitly models the transitions among individual items using a simple, efficient, and effective encoder-decoder based framework, denoted as `ed-Trans`. M^2 also explicitly models users' general preferences using the frequencies of items that each user has interacted with instead of the user embeddings.

We compare M^2 with 5 most recent, state-of-the-art methods on 4 public benchmark datasets in recommending the first, second and third next basket. Our experimental results demonstrate that M^2 significantly outperforms the state-of-the-art methods on all the datasets in all the tasks, with an improvement of up to 22.1%. We also conduct a comprehensive ablation study to verify the effects of the different factors. The results of the ablation study show that learning all the factors together could significantly improve the recommendation performance compared to learning each of them alone. The results also show that the encoder-decoder based `ed-Trans` in learning item transitions among baskets could outperform RNN-based methods on the benchmark datasets.

The major contributions in this paper are as follows:

- We developed a novel mixed model M^2 for next-basket recommendation. M^2 explicitly models three important factors: 1) users' general preferences, 2) items' global popularities, and 3) transition patterns among items.
- We developed a novel, simple yet effective encoder-decoder based framework `ed-Trans` to model transition patterns among items in baskets.
- M^2 significantly outperforms state-of-the-art methods. Our experimental results over 4 benchmark datasets demonstrate that M^2 achieves significant improvement in both recommending the next basket and recommending the next a few baskets, with an improvement as much as 22.1%. Our ablation study shows that the factors are complementary and enable better performance if learned together (Section 6.5).
- Our ablation study also shows that `ed-Trans` in learning item transitions among baskets could on its own significantly outperform RNN-based methods over the benchmark datasets, with an improvement as much as 25.4% (Section 6.5.2).
- Our cluster analysis shows that `ed-Trans` is able to learn similar embeddings for items which have similar transition patterns (Section 6.7).
- We discussed the potential issues of evaluation metrics, experimental protocols and settings that are typically used in next-basket recommendation, and discussed the use of a more appropriate protocol and setting in our experiments (Section 7).

- For reproducibility purposes, we released our source code and Supplementary Materials at https://github.com/ninglab/M2.

2 Related Work

2.1 Next-Basket Recommendation

Numerous next-basket recommendation methods have been developed, particularly using Markov Chains (MCs) and Recurrent Neural Networks (RNNs) etc. Specifically, MCs-based methods, such as factorized personalized Markov chains (FPMC) [2], use MCs to model the pairwise item-item transition patterns to recommend the next item or the next basket of items for each user. Wan *et al.* [8] developed factorization-based methods triple2vec and adaLoyal, in which the item-item complementarity, user-item compatibility and user-item loyalty patterns are modeled for the next-basket recommendation. Recently, RNN-based methods have been developed for the next-basket recommendation. For instance, Yu *et al.* [3] used RNNs to model users' dynamic short-term preference at different timestamps. Wang *et al.* [9] developed a hierarchical attentive encoder-decoder model, which iteratively predicts the next baskets by learning the transitions among items and leveraging both the positive and negative feedbacks from users. Hu *et al.* [1] developed an encoder-decoder RNN method Sets2Sets. Sets2Sets employs an RNN as encoder to learn users' dynamic preference at different timestamps and another RNN as decoder to generate the recommendation score from the learned preferences for each recommendation candidate. Sets2Sets has been demonstrated as the state of the art, and outperforms an extensive set of existing methods.

Aside from model-based methods, popularity-based approaches such as popularity on people (POP) [1] and popularity on each person (POEP) [1], are also recently employed for the next-basket recommendation. POP ranks items based on their popularity among all the users and recommend the top-k most popular items to each user. POEP is the personalized version of POP. It ranks items based on their popularity of each user and recommends the top-k most popular items of each user. These two popularity-based methods have been demonstrated as strong baselines on the next-basket recommendation in the recent work [1].

Unlike existing RNN-based approaches, M^2 does not use the Markov chains or complicated RNNs to model the transitions among items, or generate embeddings for users. Instead, M^2 models the transitions among items using a simple yet effective fully-connected layer, and explicitly models users' general preferences as the frequencies of items that users have interactions with. Our experimental results demonstrate the superior performance of M^2 over the state-of-the-art baseline methods. Our ablation study also shows that the fully-connected layer is more effective than RNNs in terms of the recommendation performance.

2.2 Sequential Recommendation

Sequential recommendation is to generate the recommendation of the next items based on users' historical interactions as in a sequence. This task is closely related to the next-basket recommendation. Please refer to Section S1 in the Supplementary Materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/

10.1109/TKDE.2022.3142773[1] for a detailed discussion about the relations between these two tasks. The sequential recommendation methods focus on capturing the sequential dependencies among individual items instead of baskets. In the last few years, numerous sequential recommendation methods have been developed, particularly using neural networks such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and attention or gating mechanisms, etc. RNN-based methods such as User-based RNN [10] explicitly integrates user characteristics into gated recurrent units (GRUs) for personalized recommendation. Skip-gram-based methods such as item2vec [11] and prod2vec [12] leverage the skip-gram model [13] to learn transition patterns among individual items. Recently, CNN-based and attention-based methods have been developed for sequential recommendation. For example, Tang *et al.* [14] developed a convolutional sequence embedding recommendation model (Caser), which uses convolutional filters on the most recent items to extract union-level features. Kang *et al.* [15] developed a self-attention based sequential model (SASRec), which uses attention mechanisms to capture the most informative items in users' historical interactions to generate recommendations. Sun *et al.* [16] further developed a bidirectional self-attention based sequential model (BERT4Rec), which employs a bidirectional attention mechanism to better model users' historical interactions. Recently, Ma *et al.* [17] developed a hierarchical gating network (HGN), which uses gating mechanisms to identify important items and generate recommendations. Peng *et al.* [18] developed hybrid associations models (HAM), which adapt the pooling mechanisms to model the association patterns and synergies among items.

2.3 Session-Based Recommendation

Session-based recommendation seeks to generate the recommendations of the next items in the current session or future sessions based on users' interactions in historical sessions. This task is also closely related to the next-basket recommendation. The session-based recommendation methods focus on capturing the intra- or inter-session dependencies to generate the recommendations [19]. In the last few years, neural networks such as RNNs, attention mechanisms and graph neural networks (GNNs) are employed in developing session-based recommendation methods. RNN-based methods such as GRU4Rec [20] and GRU4Rec+ [21] employ gated recurrent units (GRUs) to capture the users' dynamic short-term preferences over sessions. Attention-based methods such as NARM [22] and STAMP [23] employ attention mechanisms to identify the important items in recent sessions to capture users' short-term preferences. Recently, GNN-based methods have also been developed for the session-based recommendation. For example, Wu *et al.* [24] developed a GNN-based recommendation model (SR-GNN) to better model the long-term dependency among sessions. Qiu *et al.* [25] re-examined the item ordering in session-based recommendations and developed a GNN-based model (FGNN) to identify the items representing users' short-term preferences in sessions.

[1]Section references starting with "S" refer to the sections in the Supplementary Materials, available online.

3 Definitions and Notations

In this paper, the historical interactions (e.g., purchases, check-ins) of the i-th user in chronological order are represented as a sequence of baskets $B_i = \{b_i(1), b_i(2), \cdots\}$, where $b_i(t)$ is a basket of one or more items in the t-th interaction. Note that there may be multiple, same items in each basket. The number of baskets in B_i and the number of items in $b_i(t)$ is denoted as T_i and $n_i(t)$, respectively. In this paper, we consider all the baskets in users' history and all the items in each basket. We do not have a predefined maximum length for the basket sequences, and maximum size for each basket. When no ambiguity arises, we will eliminate i in $B_i/b_i(t)$, T_i and $n_i(t)$. In this paper, all the vectors are by default row vectors and represented using lower-case bold letters; all the matrices are represented using upper-case letters. The key notations are in Table 1.

4 Methods

4.1 Modeling Important Factors in M^2

M^2 has three variants M^2-p^2, M^2-gp^2 and M^2-gp^{2}t. Fig. 1 presents the M^2-p^2 and M^2-gp^2 models. Fig. 2 presents the M^2-gp^{2}t model. In these figures, each input basket is represented as a vector of n (i.e., the number of items) dimensions, in which the value in each dimension represents the number of the corresponding item in this basket. M^2 generates recommendations for the next baskets of items for each user using three factors: 1) users' general preferences, 2) items' global popularities and 3) the transition patterns among items across baskets. These three factors will be used to calculate a recommendation score for each candidate item in the next baskets. In this section, we will first describe how these three factors are modeled. In the next Section, we will describe how the three variant methods use these factors for recommendations.

4.1.1 Modeling Users' General Preferences (UGP)—Previous studies have shown that users' interactions are significantly affected by their general preferences [1], [6], which are also known as the long-term preferences in the literature [6], [18]. For example, some users prefer items of low price, while others may like luxurious items that could be expensive. Therefore, we explicitly model the general preferences of users, denoted as UGP, in M^2. Existing methods [2] usually model users' general preferences using the embeddings of users. However, there is limited, if any, validation showing that the learned embeddings could accurately capture users' preferences and to what extent. Thus, in M^2, we propose to use the frequencies of items that each user has interactions with to represent users' general preferences. The intuition is that if a user has many interactions with an item, the user has a high preference on the item and the item represents the user's preference.

Specifically, given each user's historical interactions, her/his general preference is represented as follows,

$$\mathbf{p} = [p_1, p_2, \cdots, p_n] \in \mathbb{R}^{1 \times n}, \quad (1)$$

where

$$p_j = n_j / \sum_j n_j, \tag{2}$$

n is the total number of unique items among all the baskets, and n_j is the total number of interactions with item j of the user among all her/his interactions, and thus $p_j \geq 0$, $\Sigma_j p_j = 1$. Here, we do not weight the interactions on items differently based on when they occur. This is because in real applications, we typically only use the data in the past, relatively short period of time (e.g., a few months) to train recommendation models [26]. In this short period, we can assume that most of the users will not change their general preferences dramatically, and thus all their interacted items will contribute to their `UGP` estimation evenly. A distinct advantage of the preference representation UGP as in Equation (1) compared to embedding representations for user preferences is that the `UGP` representation is very intuitive and easy to validate, and loses minimum user information.

The formulation of users' general preferences in Equation (1) is designed for the application scenarios that users are likely to have multiple interactions with the same item (e.g., online shopping, grocery shopping). For the other few application scenarios that do not have this property (e.g., movie recommendation), our formulation may not be applicable. We leave the investigation of these applications in the future work.

4.1.2 Modeling Items' Global Popularities (`IGP`)—

It has been shown in the literature [1], [2], [27], [28] that the items' global popularities also significantly influence users' purchases. Specifically, users may prefer popular items than those non-popular ones due to the herd behaviors [29], that is, they prefer to purchase items that are also purchased by many others. In $\mathtt{M}^2$, the items' global popularities are represented as in the following vector **v**,

$$\mathbf{v} = [v_1, v_2, \cdots, v_n] \in \mathbb{R}^{1 \times n}, \tag{3}$$

where n is the total number of unique items among all the baskets, and v_j is a learnable scalar to represent the global popularities of item j. Intuitively, if item j is popular, v_j will be large. Here, following the ideas in Koren *et al.* [27], we learn the popularity representations (i.e., v_j) for items via learning and optimizing from data for better performance rather than directly calculating them from data.

4.1.3 Modeling Transitions Among Items (`TPI`) via an Encoder-Decoder Framework (`ed-Trans`)—

The transitions among items is another important factor in inducing the next baskets of items that the users will be interested in [2], [14], [18]. For example, if a user purchased cat toys in a basket, she/he is likely to purchase cat food and treats in the next baskets compared to wine and beers, as there could be stronger transitions among cat items compared to from cat items to alcohols. In $\mathtt{M}^2$, we explicitly model the item transitions, denoted as `TPI`, and their effects on the next baskets. Specifically, we model the item transitions via an encoder-decoder based framework, denoted as `ed-Trans`, which takes the individual items in the historical interactions as input to predict the items in the next baskets.

`ed-Trans` Encoder.: We first represent the items aggregated over all the baskets of each user using a vector **g**

$$\mathbf{g} = [g_1, g_2, \cdots, g_j, \cdots, g_n] \in \mathbb{R}^{1 \times n}, \tag{4}$$

where g_j is the total number of interactions with item j of the user among all her/his baskets, weighted by a time-decay parameter

$$g_j = \sum_{t=1}^{T} \gamma^{T-t} \mathbb{1}(\text{item } j \in b(t)), \tag{5}$$

where $\gamma \in (0, 1]$ is the time-decay parameter to emphasize the items in the most recent baskets more than those in early baskets, and $\mathbb{1}(x)$ is an indicator function ($\mathbb{1}(x) = 1$ if x is is true, otherwise 0). Existing methods usually use RNNs to learn weights for different baskets. However, recommendation datasets are always super sparse so that RNNs may not learn meaningful weights in such sparse datasets. Instead, in $\mathtt{M}^2$, we leverage the fact that the recent interacted items affect the next basket of items more significantly compared to the items interacted much earlier [14], [15], and use the time-decay factor γ to explicitly assign and incorporate the different weights.

Given $\boldsymbol{g}$, we use a simple fully-connected layer as the encoder to encode the hidden representation of the next basket $\mathbf{h} \in \mathbb{R}^{1 \times d}$ as follows:

$$\mathbf{h} = \tanh(\mathbf{g}W), \tag{6}$$

where $W \in \mathbb{R}^{n \times d}$ is a learnable weight matrix and `tanh()` is the non-linear hyperbolic tangent activation function. Thus, the fully-connected layer represents the transition from all previous items to the items in the next basket. Here, we do not explicitly normalize **g** because the learnable parameter W will accommodate the normalization. Different from RNN-based methods which learn the transition patterns in a recurrent fashion and update the hidden states sequentially at each time stamp, the aggregation through the fully-connected layer in Equation (6) can be done much more efficiently as the item representation in Equation (4) can be done within a map-reduce framework [30] and thus in parallel. Therefore, `ed-Trans` could be more efficient than RNN-based methods especially on modeling long interaction sequences.

`ed-Trans` Decoder.: Given **h**, we use a fully-connected layer as the decoder to decode the recommendation scores **s** for all the item candidates in the next basket as follows:

$$\mathbf{s} = \text{softmax}(\mathbf{h}A + \mathbf{b}), \tag{7}$$

where $\mathbf{s} \in \mathbb{R}^{1 \times n}$ is a vector in which the j-th dimension has the recommendation score of item j, $A \in \mathbb{R}^{d \times n}$ is a learnable matrix and **b** is a bias vector. The bias vector can also be interpreted as the items' global popularities because it is shared among all the baskets. Thus, `ed-Trans` could capture both the transition patterns and items' global popularities.

4.2 Calculating Recommendation Scores in $\mathtt{M}^2$

4.2.1 Recommendation Scores Using UGP and IGP—We propose a variant of $\mathtt{M}^2$ to generate recommendations by combining the representations of users' general preferences **p** and items' global popularities **v** only. This method is referred to as mixed models with preferences and popularities and denoted as $\mathtt{M}^2\text{-}\mathtt{p}^2$. In $\mathtt{M}^2\text{-}\mathtt{p}^2$, the recommendation scores of item candidates are calculated as follows:

$$\hat{\mathbf{r}} = (1 - \alpha)\mathbf{p} + \alpha \ \mathrm{softmax}(\mathbf{v}), \tag{8}$$

where $\hat{\mathbf{r}} \in \mathbb{R}^{1 \times n}$ is the vector of recommendation scores, and α is a learnable weight to model the importance of users' general preferences and items' global popularities in users' interactions. The softmax function is employed to normalize **v** to be in the same range with **p**. The intuition here is that, as shown in the literature [1], [2], users' general preferences and items' global popularities significantly affect users' interactions. Thus, combing these two important factors should lead to reasonable recommendations. Based on the scores, the items with the top-k largest scores will be recommended into the next basket.

In $\mathtt{M}^2\text{-}\mathtt{p}^2$, in principle, α could be modeled as a tunable parameter or a learnable weight. To be consistent with the other $\mathtt{M}^2$ variants that will be presented in Sections 4.2.2 and 4.2.3, and to optimize performance, we model α as a learnable weight, and learn it in an end-to-end fashion.

4.2.2 Recommendation Scores Using Gating Networks—One possible limitation of $\mathtt{M}^2\text{-}\mathtt{p}^2$ could be that in $\mathtt{M}^2\text{-}\mathtt{p}^2$, we use a single weight α for all the users. In this way, $\mathtt{M}^2\text{-}\mathtt{p}^2$ can not capture the pattern that the weight could be different on different users. To resolve this limitation, we follow the idea of gating networks [17] to calculate personalized weight α. Specifically, we calculate the α using **p** (Equation (1)) and **v** (Equation (3)) as follows:

$$\alpha = \sigma\left(\mathbf{p}\mathbf{c}^\top + \mathbf{v}\mathbf{q}^\top\right), \tag{9}$$

where $\sigma()$ is the sigmoid function, $\mathbf{c}^\top$ and $\mathbf{q}^\top$ are learnable weight vectors. The intuition here is that the importance of UGP and IGP (i.e., α) would be learned from themselves (i.e., **p** and **v**). The method with personalized weights is referred to as mixed models with gated preferences and popularities, denoted as $\mathtt{M}^2\text{-}\mathtt{gp}^2$.

4.2.3 Recommendation Scores Using UGP, IGP and TPI—Considering all the three important factors, we propose a unified method with preferences, popularities and transitions, denoted as $\mathtt{M}^2\text{-}\mathtt{gp}^2\mathtt{t}$. In $\mathtt{M}^2\text{-}\mathtt{gp}^2\mathtt{t}$ we calculate the recommendation scores vector, $\hat{\mathbf{r}} \in \mathbb{R}^{1 \times n}$ using the representation **p** (Equation (1)) generated from UGP and the recommendation scores **s** (Equation (7)) from ed-Trans as follows:

$$\hat{\mathbf{r}} = (1 - \alpha)\mathbf{p} + \alpha\mathbf{s}, \tag{10}$$

where, similarly with that in $\mathtt{M}^2\text{-}\mathtt{gp}^2$, α is calculated from **p** (Equation (1)) and **h** (Equation (6)) as following:

$$\alpha = \sigma\left(\mathbf{p}\mathbf{c}^{\top} + \mathbf{h}\mathbf{q}^{\top}\right), \tag{11}$$

where, as presented in Section 4.2.2, $\sigma()$ is the sigmoid function, $\mathbf{c}^{\top}$ and $\mathbf{q}^{\top}$ are learnable weight vectors. Please note that as discussed in Section 4.1.3.2, the scores in **s** are generated using both items' popularities and the transition patterns. Thus, `M`^{2}`-gp`^{2}`t` uses all the three factors to make recommendations. Also note that, as shown in Equation (7), the vector **s** is already normalized to be in the same range with **p**. Therefore, we do not need the softmax function for the normalization in Equation (10).

4.3 Network Training

We minimize the negative log likelihood that the ground-truth items in the next baskets have high recommendation scores. The optimization problem is formulated as follows,

$$\min_{\boldsymbol{\Theta}} \sum_{i=1}^{m} -\mathbf{r}_i \log\left(\hat{\mathbf{r}}_i^{\top}\right) + \lambda \parallel \boldsymbol{\Theta} \parallel^2, \tag{12}$$

where m is the number of users to recommend baskets to, $\mathbf{r}_i$ and $\hat{\mathbf{r}}_i$ are for the i-th user, $\boldsymbol{\Theta}$ is the set of the parameters, and λ is the regularization parameter. Following previous work [1], [3], we calculate the training error on the last basket in training data. The vector $\mathbf{r}_i$ is the vector representation of the items in the last basket $b_i(T)$, in which the dimension j is 1 if item j is in $b_i(T)$ or 0 otherwise. Here, we do not consider the frequencies of individual items in the baskets (i.e, $\mathbf{r}_i$ is binary), as we do not predict the frequencies of items in the next baskets. We optimize Problem 12 using the Adagrad optimization method [31]. The parameter tuning protocol and all the parameters for modeling are reported in Section S3.

5 Experimental Settings

5.1 Baseline Methods

We compare `M`^{2} with 5 state-of-the-art baseline methods on next-basket recommendations: 1) `POP` [1] ranks items based on their popularity among all the users, and recommends the top-k most popular items. 2) `POEP` [1] ranks items based on their popularity on each user and recommends the personalized top-k most popular items. 3) `Dream` [3] uses RNNs to model users' preferences over time. It uses the most recent hidden state of RNNs to generate recommendation scores and recommends the items with top-k scores. 4) `FPMC` [2] models users' long-term preferences and the transition patterns of items using the first-order markov chain and matrix factorization. 5) `Sets2Sets` [1] adapts the encoder-decoder RNNs to model the short-term preferences and the recurrent behaviors of users. Please note that `Sets2Sets` achieves the state-of-the-art performance on the next-basket recommendation and outperforms other methods [2], [3], [32]. Therefore, we compare `M`^{2} with `Sets2Sets` but not the methods that `Sets2Sets` outperforms.

5.2 Datasets

We generate 4 datasets from 3 benchmark datasets TaFeng[2], TMall[3], and Gowalla[4] to evaluate the different methods. TaFeng has grocery transactions in 4 months (i.e., 11/1/2020 to 02/28/2020) at a grocery store and each basket is a transaction of grocery items. TMall has online transactions in 5 months (i.e., 07/01/2015 to 11/31/2015) and each basket is a transaction of products. Gowalla [33] is a place-of-interests dataset and contains user-venue check-in records with timestamps. Similarly to Ying *et al.* [34], we view the check-in records in one day as a basket and focus on the records in 10 months (i.e., 01/01/2010 to 10/31/2010).

Following previous work [34], we do the following filtering to generate the datasets we will use in the experiments: 1) filter out the infrequent users with fewer than 10, 20 and 15 items from the original TaFeng, TMall and Gowalla dataset, respectively, 2) filter out infrequent items interacted by fewer than 10, 20 and 25 users from the TaFeng, TMall and Gowalla dataset, respectively, and 3) filter out users with fewer than 2 baskets. Out of the above three filtering steps, each of the 3 original datasets will have frequent users and items, and we denote the processed datasets still as TaFeng, TMall and Gowalla. In order to better evaluate the methods in real applications that have a large amount of users and items, from the original TMall dataset, we also apply a smaller threshold 10 on user frequency and item frequency to generate another dataset, denoted as sTMall, with more users and items retained. The statistics of the preprocessed datasets are presented in Table 2. We noticed that the Dunnhumby dataset[5] and the Instacart dataset[6] are also used in the literature [1], [8]. However, Dunnhumby is a simulated dataset and the Instacart dataset is not publicly available now. Therefore we do not use these datasets in our experiments. We discussed the limitations of these datasets in detail in Section S6.

5.3 Experimental Protocol

Similarly to Ying *et al.* [34], we split the 4 datasets based on cut-off times as shown in Fig. 3. Specifically, on TaFeng, we use the transactions in the first 3 months as the training set, the transactions in the following 0.5 month as the validation set, and the transactions in the last 0.5 month as the testing set. Similarly, on Gowalla, we use the records in the first 8 months as the training set, the records in the following 1 month as the validation set, and the records in the last 1 month as the testing set. On TMall and sTMall, we use the transactions in the first 3.5 months as the training set, the transactions in the following 0.5 month as the validation set, and the transactions in the last 1 month as the testing set. We split the datasets in this way to guarantee that all the interactions in the testing set occur after the interactions in the training and validation sets. Thus, the setting is close to real use scenarios. A detailed discussion about different experimental protocols is presented later in Section 7.1.

We denote the baskets in the training, validation and testing sets as training, validation and testing baskets, respectively. The users which have interactions in the training, validation and testing sets are denoted as training, validation and testing users, respectively. Please note that a user can be both training and testing user if she/he has baskets in both training and testing sets. During training, we only use the interactions in the training baskets to estimate users'

2. https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset
3. https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
4. https://snap.stanford.edu/data/loc-Gowalla.html
5. https://www.dunnhumby.com/source-files/
6. https://www.instacart.com/datasets/grocery-shopping-2017

general preferences and to learn item transition patterns. There could be items in testing or validation baskets that never appeared in training baskets (i.e., cold-start items). In this case, we will retain the baskets with such items. Since $\mathtt{M}^2$ and all the baseline methods are not developed for the cold-start problem [35], the cold-start items will not get recommended but the baskets with such items can still be evaluated due to other items.

We tune the parameters using grid search and use the best parameters in terms of recall@5 on the validation set during testing for the $\mathtt{M}^2$ and all the baseline methods. Following previous work [14], [15], [17], during testing, we use the interactions in both training and validation sets to train the model with the optimal parameters identified at the validation set. Similarly to Hu *et al.* [1], we evaluate $\mathtt{M}^2$ and baseline methods on three tasks: recommending the first next basket, the second next basket and the third next basket. Please note that in recommending the second next or third next basket, during evaluation, the first or second testing basket, respectively, of testing users will be used to update the user's general preference representation **p** (Equation (1)) and item transitions in **g** (Equation (4)). Also note that the number of validation and testing users in these three tasks could be different. In recommending the second next basket, only users with at least two validation or testing baskets are used as validation or testing users, but users with only one validation or testing basket will not be used in evaluation.

5.4 Evaluation Metrics

We use recall@k, precision@k, and NDCG@k to evaluate the different methods. For each user, recall measures the proportion of all the ground-truth interacted items in a testing basket that are correctly recommended. We denote the set of k recommended items and the set of the items in the ground-truth basket as R_k and S, respectively. Given R_k and S, recall@k is calculated as follows:

$$\text{recall}@k = \frac{|R_k \cap S|}{|S|}, \tag{13}$$

where $R_k \cap S$ is the intersection between the two sets and $|S|$ denotes the size of the set S. Precision measures the proportion of all the recommended items that are correctly recommended, and precision@k is calculated as follows:

$$\text{precision}@k = \frac{|R_k \cap S|}{k}. \tag{14}$$

We report in the experimental results the recall@k and precision@k values that are calculated as the average over all the testing users. Higher recall@k and precision@k indicate better performance. It is worth noting that although we use precision@k in our experiments, we argue that precision@k may not be a proper metric for evaluating next-basket recommendation methods as we will discuss later in Section 7.2.

NDCG@k is the normalized discounted cumulative gain for the top-k ranking. In our experiments, the gain indicates whether a ground-truth item is recommended (i.e., gain is 1) or not (i.e., gain is 0). NDCG@k incorporates the positions of the correctly recommended items among the top-k recommendations. Higher NDCG@k indicates the ground-truth items are recommended at very top, and thus better recommendation performance.

Besides these evaluation metrics, we also statistically test the significance of the performance difference among different methods via a standard t-test. Specifically, we conducted t-test over the paired recall, NDCG and precision values from different methods. If the p-values are smaller than a predefined threshold α (α = 0.05 in our experiments), the performance difference of two methods is considered statistically significant at 100(1-a)% confidence level.

6 Experimental Results

6.1 Overall Performance on the First Next Basket

Table 3 presents the overall performance at recall@k and NDCG@k in recommending the first next basket of all the methods on the 4 datasets. Due to the space limit, we report the performance at precision@k in Section S4.1. In Table 3, for each dataset, the best performance among `M`^{2} variants (i.e., `M`^{2}`-p`^{2}, `M`^{2}`-gp`^{2} and `M`^{2}`-gp`^{2}`t`) is in bold, the best performance among baseline methods (e.g., `POP`, `POEP`, `Sets2Sets`) is underlined. and the overall best performance is indicated by a dagger (i.e., †). We report the parameters that achieve the reported performance also in Section S3. For `Sets2Sets`, we use the implementation provided by the authors. However, this implementation raises memory issues and cannot fit in 16GB GPU memory on the largest dataset sTMall. Therefore, we report out of memory (OOM) for `Sets2Sets` on sTMall.

Table 3 shows that overall, `M`^{2}`-gp`^{2}`t` is the best performing method on the task of recommending the first next basket. In terms of recall@5, recall@10 and recall@20, `M`^{2}`-gp`^{2}`t` achieves the best performance with significant improvement compared to the second best method on TaFeng and sTMall. On the TMall and Gowalla datasets, `M`^{2}`-gp`^{2}`t` also achieves the best or second best performance at recall@5, recall@10 and recall@20. Compared to the second best method, `M`^{2}`-gp`^{2}`t` achieves on average 6.8%, 2.9% and 2.5% improvement at recall@5, recall@10 and recall@20, respectively, over all the datasets. In terms of NDCG@5, NDCG@10 and NDCG@20, `M`^{2}`-gp`^{2}`t` achieves the best performance on TaFeng, sTMall and Gowalla, and the second best performance on the TMall dataset. In particular, on the largest dataset sTMall, `M`^{2}`-gp`^{2}`t` achieves substantial improvement of 10.8% on average over all the metrics compared to the second best method. On the most widely used benchmark dataset TaFeng, `M`^{2}`-gp`^{2}`t` also achieves significant improvement of at least 2.3% over the second best method at all the metrics. On Gowalla where many baseline methods do not perform well, `M`^{2}`-gp`^{2}`t` is still slightly better than the second best method `M`^{2}`-gp`^{2}. It is also worth noting that, compared to the performance of the best baseline methods (underlined in Table 3), `M`^{2}`-gp`^{2}`t` achieves statistically significant improvement over most of the metrics on 3 out of 4 datasets. On the TMall dataset, `M`^{2}`-gp`^{2}`t` still achieves statistically significant improvement over the best baseline methods at both recall@5 and recall@10. These results demonstrate the strong performance of `M`^{2}`-gp`^{2}`t`. `M`^{2}`-gp`^{2} is the second best performing method in our experiments. It achieves the second best or (near) the second best performance on all the four datasets. We notice that POP, Dream and FPMC work poorly on the Gowalla dataset. This might be due to the fact that these methods do not really capture the personalized general preferences of users. Recall that the Gowalla dataset is a place-of-interests dataset, which contains user-venue check-in records of users. Different users live in different places and could interact with very different items. Thus, methods without explicitly model users' personalized general preferences could not work well on this dataset.

6.1.1 Comparing `M2-gp2t` With Model-Based Methods—

Table 3 also shows that among the 4 model-based methods `Dream`, `FPMC`, `Sets2Sets` and $\texttt{M}^2\texttt{-gp}^2\texttt{t}$, $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ consistently and significantly outperforms `Dream` and `FPMC` on all the datasets. The primary difference among $\texttt{M}^2\texttt{-gp}^2\texttt{t}$, `Dream` and `FPMC` is that $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ explicitly models users' general preferences using the frequencies of the items that each user has interactions with, while `Dream` and `FPMC` implicitly model them using the hidden state of RNNs or user embeddings. Given the sparse nature of recommendation datasets (Table 2), it is possible that the learned hidden states or user embeddings cannot represent the user preferences well, as the signals of user preferences are smoothed out due to data sparsity during the recurrent updates, or by the pooling or weighting schemes used to learn user embeddings as some other work also noticed [18], [36], [37]. The superior performance of $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ over `Dream` and `FPMC` on all the datasets demonstrates the effect of explicitly modeling users' general preferences.

Table 3 shows that $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ significantly outperforms `Sets2Sets` on all the datasets except TMall in terms of both recall@k and NDCG@k. The primary differences between $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ and `Sets2Sets` are 1) $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ explicitly models the transition patterns among items using encoder-decoder-based `ed-Trans`, while `Sets2Sets` implicitly models the transition patterns using RNNs, and 2) when calculating the recommendation scores, $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ learns a single weight on each user (i.e., α in Equation (10)), but `Sets2Sets` learns different weights for different items on each user. Given the sparse nature of the recommendation datasets, weights for different items on each user may not be well learned [18], [36]. Thus, such weights may not necessarily help better differentiate user general preferences over items. In addition, the learned weights over items may guide the model to learn inaccurate general preferences of users, and thus degrade the performance. We also notice that on TMall, $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ underperforms `Sets2Sets` in terms of NDCG but outperforms `Sets2Sets` in terms of recall. This indicates that on certain datasets, $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ could be more effective than `Sets2Sets` on ranking the items of users' interest on top of the recommendation list, while less effective than `Sets2Sets` on raking these items on the very top. However, `Sets2Sets` is very memory consuming, demonstrated by out of memory (OOM) issues on the largest dataset sTMall, which substantially limits its use in real, large-scale recommendation problems.

6.1.2 Comparing `M2-gp2t` With Popularity-Based Methods—

In Table 3, we also notice that $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ statistically significantly outperforms the best popularity-based method $\texttt{M}^2\texttt{-gp}^2$ on all the datasets. On average, it achieves 6.8%, 3.0%, 9.3% and 7.8% improvement over $\texttt{M}^2\texttt{-gp}^2$ in terms of recall@5, recall@10, NDCG@5 and NDCG@10, respectively, over all the datasets. Recall that the key difference between $\texttt{M}^2\texttt{-gp}^2\texttt{t}$ and

$\mathtt{M}^2$-$\mathtt{gp}^2$ is that $\mathtt{M}^2$-$\mathtt{gp}^2\mathtt{t}$ models users' general preferences, items' global popularities and the transition patterns, whereas $\mathtt{M}^2$-$\mathtt{gp}^2$ only models users' general preferences and items' global popularities. These results demonstrate the importance of transition patterns in sequence-based next-basket recommendation.

6.1.3 Comparison Among Popularity-Based Methods—

Among the four popularity-based methods POP, POEP, $\mathtt{M}^2$-$\mathtt{p}^2$ and $\mathtt{M}^2$-$\mathtt{gp}^2$, $\mathtt{M}^2$-$\mathtt{gp}^2$ achieves the best performance at most of the metrics on all the 4 datasets. Between $\mathtt{M}^2$-$\mathtt{gp}^2$ and $\mathtt{M}^2$-$\mathtt{p}^2$, $\mathtt{M}^2$-$\mathtt{gp}^2$ outperforms $\mathtt{M}^2$-$\mathtt{p}^2$ on the TaFeng and Gowalla datasets, and achieves similar performance with $\mathtt{M}^2$-$\mathtt{p}^2$ on the TMall and sTMall datasets. In terms of recall@5, $\mathtt{M}^2$-$\mathtt{gp}^2$ outperforms $\mathtt{M}^2$-$\mathtt{p}^2$ on all the datasets. In terms of recall@10 and recall@20, $\mathtt{M}^2$-$\mathtt{gp}^2$ outperforms $\mathtt{M}^2$-$\mathtt{p}^2$ on the TaFeng and sTMall datasets, and achieves similar performance with $\mathtt{M}^2$-$\mathtt{p}^2$ on the TMall and Gowalla datasets. We also found a similar trend on NDCG@*k*: for example, in terms of NDCG@5, $\mathtt{M}^2$-$\mathtt{gp}^2$ outperforms $\mathtt{M}^2$-$\mathtt{p}^2$ on all the datasets except sTMall. On sTMall, $\mathtt{M}^2$-$\mathtt{gp}^2$ achieves the same performance with $\mathtt{M}^2$-$\mathtt{p}^2$. The difference between $\mathtt{M}^2$-$\mathtt{gp}^2$ and $\mathtt{M}^2$-$\mathtt{p}^2$ is that $\mathtt{M}^2$-$\mathtt{gp}^2$ learns personalized weights to combine users' general preferences and items' global popularities, while $\mathtt{M}^2$-$\mathtt{p}^2$ only learns one such weight for all the users. The substantial performance improvement of $\mathtt{M}^2$-$\mathtt{gp}^2$ over $\mathtt{M}^2$-$\mathtt{p}^2$ demonstrates the importance of learning personalized weights. We also notice that overall, $\mathtt{M}^2$-$\mathtt{p}^2$ consistently outperforms POP and POEP on all the datasets over all the metrics. In terms of recall@5, recall@10 and recall@20, $\mathtt{M}^2$-$\mathtt{p}^2$ consistently outperforms POP and POEP at all the 4 datasets. For example, on the widely used TaFeng dataset, in terms of recall@10, $\mathtt{M}^2$-$\mathtt{p}^2$ achieves significant improvement of 38.9% and 16.6% compared to POP and POEP, respectively. Recall that the difference between $\mathtt{M}^2$-$\mathtt{p}^2$, POP and POEP is that $\mathtt{M}^2$-$\mathtt{p}^2$ models both users' general preferences and items' global popularities, while POP and POEP only model one of them. The substantial improvement of $\mathtt{M}^2$-$\mathtt{p}^2$ over POP and POEP demonstrates that items' global popularities and users' general preferences are complementary. When learned together, they will enable better performance than each alone. It is also worth noting that $\mathtt{M}^2$-$\mathtt{gp}^2$ outperforms the state-of-the-art model-based method Sets2Sets at all the metrics on TaFeng and Gowalla. The superior performance of $\mathtt{M}^2$-$\mathtt{gp}^2$ is a strong evidence that the simple popularity-based methods could still be very effective in next-basket recommendations.

6.2 Performance on the Second Next Basket

Table 4 presents the overall performance of different methods at recall@*k* and NDCG@*k* in recommending the second next basket (i.e., the second basket in the testing set) on the 4 datasets. We also report the performance at precision@*k* in Section S4.2. The parameter tuning protocol in this task is the same as that in recommending the first next basket (Section 6.1). As discussed in Section 5.3, when recommending the second next basket, the first testing basket of users will be used to update the models. In addition, in this task, only users with at least two testing baskets will be used as testing users. Thus, the number of testing users in this task could be different from that in recommending the first next basket. Specifically, as shown in Table 4, when recommending the second next basket, we have

2,801, 5,109, 29,741 and 10,032 testing users on TaFeng, TMall, sTMall and Gowalla, respectively.

6.2.1 Overall Performance—As shown in Table 4, overall, in recommending the second next basket, the performance of M^2 and baseline methods has a similar trend as that in recommending the first next basket. In particular, M^2-gp^{2}t is still the best performing method in this task. In terms of recall@5, M^2-gp^{2}t achieves the best performance on all the 4 datasets. In terms of recall@10, M^2-gp^{2}t achieves the best performance on the sTMall and Gowalla datasets, and the second best performance on the TaFeng and TMall datasets. We also found a similar trend on NDCG@*k*: in terms of NDCG@5, M^2-gp^{2}t achieves the best performance on the sTMall and Gowalla datasets, and the second best or (near) the second best performance on the TaFeng and TMall datasets. M^2-gp^2 is still the second best performing method. In terms of recall@5 and recall@10, M^2-gp^2 achieves the best performance on the TaFeng dataset, and the second best performance or (near) the second best performance on the other 3 datasets (i.e., TMall, sTMall, Gowalla). In terms of NDCG@5 and NDCG@10, M^2-gp^2 also achieves the second best or (near) the second best performance on 3 out of 4 datasets (i.e., TaFeng, sTMall, Gowalla). It is also worth noting that on the widely used TaFeng dataset, M^2-gp^2 significantly outperforms M^2-p^2 at 24.6%, 13.3%, 16.0% and 13.4% on recall@5, recall@10, NDCG@5 and NDCG@10, respectively. As discussed in Section 6.1.3, the difference between M^2-gp^2 and M^2-p^2 is that M^2-gp^2 learns personalized combine weights, while M^2-p^2 learns one combine weight for all the users. The significant improvement of M^2-gp^2 over M^2-p^2 further demonstrates the importance of learning personalized combine weights.

6.2.2 Comparing With the Performance on the Next Basket—We also notice that the performance of those methods that model users' general preferences (e.g., POEP, M^2-gp^2 and M^2-gp^{2}t) increases as we recommend the baskets in the later future (i.e., the second next basket). For example, on the largest sTMall dataset, POEP has recall@5 value 0.0936 (Table 3) in recommending the first next basket, while this value increases to 0.1132 (Table 4) in recommending the second next basket. This might be due to the fact that the testing users with more than one basket in the testing set are in general more active (i.e., have more baskets). Specifically, on sTMall, the testing users in the experiments of recommending the first next basket have 9.6 baskets on average used for training. However, the testing users in the experiments of recommending the second next basket have 10.6 baskets on average (i.e., 10.4% increasing). Thus, more baskets used for model training enable methods which model users' general preferences to more accurately estimate the general preferences of testing users, and thus achieve better performance for the second next basket recommendation.

It is worth noting that although M^2-gp^2 significantly underperforms M^2-gp^{2}t when recommending the first next basket, M^2-gp^2 could achieve similar or even better performance over M^2-gp^{2}t at some metrics when recommending the second next basket. For example, on TaFeng, when recommending the first next basket, M^2-gp^{2}t achieves significant improvement of 10.6%, 8.6% at recall@5 and recall@20 (Table 3) over M^2-gp^2. However, when recommending the second next basket, M^2-gp^2 is able to achieve the same performance with M^2-gp^{2}t at recall@5 (i.e., 0.1113 as in Table 4). As just discussed, the testing users in recommending the second next basket are in general more active than those in recommending the first next basket. The similar performance of M^2-gp^2 and M^2-gp^{2}t indicates that the interactions of active users are more dominated by their general preferences and the global popularities of items. Thus, for active users, the simple popularity-based methods could be very effective. However, since in real applications, most of the users are not active, it is still important to model the transition patterns in general recommendation applications.

6.3 Performance on the Third Next Basket

Table 5 presents the overall performance of methods at recall@k and NDCG@k on the task of recommending the third next basket. The performance at precision@k is reported in Section S4.3. Please note that as discussed in Sections 5.3 and 6.2, the number of testing users in this task could be different from that in recommending the first, and second next basket. Specifically, as shown in Table 5, when recommending the third next basket, we have 1,099, 1,461, 7,561 and 7,985 testing users on TaFeng, TMall, sTMall and Gowalla, respectively. Table 5 shows that overall, the performance of M^2 and baseline methods still has similar trend as that in recommending the first and second next basket. M^2-gp^{2}t is still the best performing method. In terms of recall@5, M^2-gp^{2}t achieves the best performance at 3 out of 4 datasets (i.e., TaFeng, sTMall and Gowalla). On the TMall dataset, M^2-gp^{2}t also achieves the second best performance. We also found a similar trend on NDCG@k: in terms of NDCG@5, M^2-gp^{2}t also achieves the best performance on 3 out of 4 datasets, and the second best performance on the TMall dataset. M^2-gp^2 is still the second best performing method. In terms of recall@5, M^2-gp^2 achieves the best performance on the TMall dataset, and the second best performance on the TaFeng and sTMall dataset. The same trends as discussed in Section 6.2.1 could also be found here. It is also worth noting that as shown in Table 5, in terms of recall@k, the best M^2 variant (i.e., M^2-p^2, M^2-gp^2 or M^2-gp^{2}t), statistically significantly outperforms the best baseline methods on 3 out of 4 datasets. On TMall, M^2-gp^{2}t still statistically significantly outperforms the best baseline method POEP at recall@5.

6.4 Performance Summary Among All the Tasks

Table 3, Table 4 and Table 5 together show that M^2-gp^{2}t is the best performing method over all the 3 tasks. It significantly outperforms the state-of-the-art baseline method Sets2Sets at all the metrics over all the 3 tasks. For example, in terms of recall@5, M^2-gp^{2}t achieves 15.5%, 25.4% and 26.6% improvement on average over all the datasets except sTMall in recommending the first, second and third next basket, respectively. These results demonstrate the strong ability of M^2-gp^{2}t in next-basket recommendation. Table 3, Table 4 and Table 5 together also show that M^2-gp^2 achieves the second best performance over the 3 tasks. It is worth noting that although M^2-gp^2 does not perform as well as M^2-gp^{2}t, it still consistently outperforms the state-of-the-art baseline method Sets2Sets over all the 3 tasks. These results demonstrate the strong effectiveness of simple popularity-based methods in next-basket recommendation.

6.5 Ablation Study

6.5.1 Comparing Different Factors in M^2-gp^{2}t—We conduct an ablation study to verify the effects of the different components (i.e., UGP, TPI) in M^2-gp^{2}t. We present the next basket recommendation results generated by UGP and TPI alone, and their combination M^2-gp^{2}t in Table 6. Note that UGP recommends the personalized most popular items to each user, and thus it is identical to POEP. When testing UGP, the final recommendation scores $\hat{\mathbf{r}}$ (Equation (10)) are identical to those based on users' general preferences in $\mathbf{p}$ (Equation (1)) (i.e., α =0 in Equation (10)). When testing TPI, essentially it is to test ed-Trans and the final recommendation scores are in $\mathbf{s}$ (Equation (7)) (i.e., α =1 in Equation (10)).

Table 6 shows that UGP is a strong baseline for all the methods on all the datasets. This indicates the importance of users' general preferences in the next-basket recommendation. TPI does not outperform UGP in terms of recall@*k* on all the datasets. We also found a similar trend on NDCG@*k*. In terms of NDCG@*k*, UGP significantly outperforms TPI on TaFeng and Gowalla and achieves similar performance with TPI on the TMall and sTMall datasets. When TPI is combined with UGP (i.e., M^2-gp^{2}t in Table 6), there is a notable increase compared to each individual TPI and UGP. This may be because that in M^2-gp^{2}t, as UGP captures the general preferences, TPI can learn the remaining, transition patterns and items' global popularities that cannot be captured by UGP. In Table 6, M^2-gp^{2}t (i.e., UGP +TPI) achieves the best performance on all the 4 datasets. It also shows improvement from UGP and TPI. This indicates that when learned together, UGP and TPI are complementary and enable better performance than each alone.

6.5.2 Comparing ed-Trans and RNN-Based Methods—We also notice that as shown in Table 3 and Table 6, ed-Trans (i.e., the model to learn TPI), an encoder-decoder based approach (Section 4.1.3), on its own outperforms Dream (i.e., RNN-based method) on 3 out of 4 datasets i.e., TMall, sTMall and Gowalla) at both recall@*k* and NDCG@*k*, and achieves comparable results with Dream on the TaFeng dataset at NDCG@*k*. For example, on TMall, ed-Trans achieves 0.0947 in terms of recall@5 (Table 6) compared to Dream with 0.0833 (Table 3), that is, ed-Trans is 13.7% better than Dream. Similarly, in terms of recall@10 and recall@20, ed-Trans achieves 0.1045 and 0.1162 (Table 6), respectively, compared to Dream with 0.0868 and 0.0927 (Table 3), respectively, that is, ed-Trans achieves 20.4% improvement at recall@10 and 25.4% improvement at recall@20 compared to Dream. We also found a similar trend on sTMall. In terms of recall@5, ed-Trans achieves 8.9% improvement over Dream (0.0928 versus 0.0852) on sTMall. These results are strong evidence to show that ed-Trans could outperform RNN-based methods on benchmark datasets. It is worth noting that as shown in Table 3 and Table 6, on Gowalla, ed-Trans achieves reasonable performance, while Dream fails. As discussed in Section 6.1, for doing good recommendations on Gowalla, models should be able to learn users' general preferences from the interactions. The reasonable and poor performance of ed-Trans and Dream, respectively, indicates that ed-Trans could implicitly learn users' general preferences, while RNN-based methods might not. We also notice that ed-Trans on its own does not work as well as Sets2Sets as shown in Table 3 and Table 6. However, this might be due to the reason that Sets2Sets models both the transition patterns and users' general preferences, while `ed-Trans` does not explicitly model users' general preferences. When `ed-Trans` learned with `UGP` together (i.e, $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$), $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ outperforms `Sets2Sets` on all the datasets as shown in Table 3. These results indicate that `ed-Trans` could be more effective than the RNNs used in `Sets2Sets` on modeling transition patterns.

6.6 Analysis on Transition Patterns

We further analyze if $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ learns good weights α (Equation (10)) to differentiate the importance of `UGP` and `TPI`. Fig. 4 presents the distribution of the weights α from the best performing $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ models on the 4 datasets. Please note that as presented in Section 4.1.1, only the items interacted by the user will get non-zero recommendation scores in `UGP`, while all the items could get non-zero recommendation scores in `TPI`. As a result, for items with non-zero scores, the scale of their scores might be different in `UGP` and `TPI`. and thus, the absolute value of the weights on different components may not necessarily represent the true importance of the corresponding factors in users' behavior. For example, on TMall, users have higher weights on `TPI` than that on `UGP`. It does not necessarily indicate that the transition patterns are more important than users' general preference for the recommendation on this dataset.

As shown in Fig. 4, on Gowalla, users' weights on `UGP` are much higher than that on the other datasets. This is consistent with the observation that on Gowalla, users' general preferences play a more important role for recommendation than that on the other datasets (shown in Table 3). This consistency demonstrates that $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ is able to learn good weights to differentiate the importance of `UGP` and `TPI` on different datasets and application scenarios.

6.7 Cluster Analysis

We further evaluate if $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ really learns the transition patterns among items. Specifically, we learn the weight matrix W (Equation (6)) in $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ using the training and validation baskets in the widely used TaFeng dataset on recommending the first next basket, and export the matrix for the analysis. Note that the weight matrix W could be viewed as an item embedding matrix, in which each row is the embedding of a single item. Given W, we evaluate if items with similar transition patterns will have similar embeddings. To get the ground-truth transition patterns among items, we construct a matrix T also from the training and validation baskets in TaFeng. In T, T_{ij} is the number of times that item i in the previous baskets transits to item j in the next basket. That is, the i-th row of T contains the items that item i has transited to. Thus, T contains the ground-truth transition patterns among items. After constructing matrix T, we could get the items which have similar transition patterns by calculating the pairwise similarities.

Fig. 5, generated using t-SNE [38] method, presents the item embeddings generated from $\mathtt{M}^2\mathtt{-gp}^2\mathtt{t}$ on the TaFeng dataset. Specifically, we project the item embeddings in W to the two-dimensional (2d) space using t-SNE, and then plot the projected embeddings of items in this figure. In Fig. 5, there are many well-formed clusters (e.g., C_1, C_2). We find that generally, the items within the same cluster have similar transition patterns. For example, the average pairwise similarity of items in C_1 and C_2 is 25.7% and 11.4% higher than that over all the itempairs, respectively. These results demonstrate that the encoder-decoder framework (ed-Trans) in M^2-gp^{2}t could effectively capture the transition patterns among items.

6.8 Analysis on Diversity of Recommendations

We also evaluate the diversity of the recommendations from different methods. Due to the space limit, we report the results in Section S5. Generally, we find that M^2-gp^{2}t could generate more diverse recommendations over all the baseline methods except POEP. Considering both the quality and diversity of the recommendations, M^2-gp^{2}t significantly outperforms all the baseline methods, and could achieve superior performance in real applications.

7 Discussions

7.1 Experimental Protocols

A commonly used experimental protocol in the literature [1] is as follows. Users are randomly split into 5 or 10 folds to conduct 5 or 10-fold cross validation. For each user in the testing fold, her/his last basket in sequential order is used as the testing basket, the other baskets are used as the training baskets. For each user in the training folds, her/his last basket is used to measure training errors, the other baskets are used to train the model and generate recommendation scores for the last basket. When absolute time information is absent in the datasets, this experimental protocol enables full separation among the training and testing sets, and approximates real application scenario for each testing user. However, when the absolute time information is present, which is the case in most of the popular benchmark datasets including TaFeng, TMall and Gowalla, this protocol will create artificial use scenario that deviates from that in real applications. The issue is that following this protocol, a basket in the training set from one user may have a later timestamp than a basket in the testing set from another user, and therefore a later basket is used to train a model to recommend an earlier basket, which is not realistic. Our protocol splits the training, validation and testing sets based on an absolute cut-off time for all the users, and thus avoids the above issue and is closer to real application scenarios. Another widely used experimental protocol [2], [3], [14], [18], [39] is that for each user, her/his last and second last basket were used as the testing basket and validation basket, respectively; the other baskets are used as the training baskets. This protocol has the same issue as discussed above. Here, we refer this protocol as the order-based split protocol. We evaluate M^2 and baseline methods using this widely used but questionable order-based split protocol, and report the results in Table S1. We found that, under the order-based split protocol, M^2 still achieves superior performance over the best baseline methods on all the datasets over most of the evaluation metrics.

Another commonly used experimental setting [2], [3] is to evaluate different methods in recommending the first next basket. However, in real applications, the model is usually updated weekly or monthly, and thus would need to recommend multiple baskets for active users before model updates. In this case, the performance in recommending the first next basket may not accurately represent the models' effectiveness in real applications. In our experiments, we also evaluate methods in the task of recommending a few next baskets to more accurately and comprehensively evaluate the model performance in real applications.

7.2 Evaluation Metrics

In the experiments, we use recall@k and NDCG@k to evaluate different methods. These two metrics are important and widely used for top-N recommendation [40], and also popular in sequential recommendations [15], [18], [39] and next-basket recommendations [1], [2], [3]. Recall@k measures the proportion of all the ground-truth interacted items in a testing basket that are also among top-k recommended items. We believe this is a proper metric to use because in the end, the recommendation methods aim to identify all the items that the users will be interested in eventually, that is, to maximize recall. In addition, recall values at different top-k positions also indicate the ranking structures of recommended items, where we prefer the items that users are interested in are ranked on top. NDCG also measures the ranking positions of the items that users are interested in. Higher NDCG@k values indicate that more users' interested items are ranked on top. Since in real applications, the users will look at a subset of the recommendations from the top of the recommendation list, we believe that evaluation metrics that consider ranking positions are more useful and applicable in real applications, and as discussed in Aggarwal [40] (Chapter 7.5.5), NDCG is more suitable than ROC measures or rank-correlation coefficients in distinguishing between higher-ranked and lower-ranked items.

The metric precision@k is also a popular metric in evaluating recommendations. This metric, however, may not be proper for next-basket recommendation evaluation. First of all, precision@k does not consider the ranking positions of the correctly recommended items. Second, the value of precision@k is "not necessarily monotonic in k because both the numerator and denominator may change with k differently", as discussed in Aggarwal [40] (Chapter 7.5.4). In addition, precision@k could be strongly biased by basket sizes: for small baskets, precision@k could be small even if all the items are correctly recommended. For example, if all the items in a size-2 basket are correctly recommended, precision@10 is only 0.2. However, for large baskets, precision@k can be large even only a small portion of the items are correctly recommended. For example, if 5 items of a size-20 basket are correctly recommended, that is, only 25% of the items are correctly recommended, precision@10 is 0.5. When only considering precision@k, we may prefer the second recommendation, even thought it is half way to its best possible results (i.e., correctly recommend 10 among top-10 recommended items, with precision@10=1.0), but the first recommendation has already achieved its best possible results. Recall@k alleviates such issues with a normalization using basket size. Therefore, precision and other precisionbased metrics (e.g., AUC, F1) may not be proper for evaluating next-basket recommendation methods. However, to be comprehensive, we still use this metric in our experiments and report the results in Section S4.

8 Conclusion

In this paper, we presented novel $\mathtt{M}^2$ models that conduct next-basket recommendation using three important factors: 1) users' general preferences, 2) items' global popularities and 3) the transition patterns among items. Our experimental results in comparison with 5 state-of-the-art next-basket recommendation methods on 4 public benchmark datasets demonstrate substantial performance improvement from `M`^{2} in both the next basket recommendation (improvement of up to 19.0% at recall@5) and the next a few baskets recommendation (improvement of up to 14.4% at recall@5). Our ablation study demonstrates the importance of users' general preferences in next-basket recommendations, and the complementarity among all the factors in `M`^{2}. Our ablation study also demonstrates that the simple encoder-decoder based framework `ed-Trans` (Section 4.1.3) is more effective than RNNs on modeling the transition patterns in benchmark datasets (improvement as much as 20.4% at recall@5). Our analysis on the learned item embedding matrix further demonstrates that `ed-Trans` could effectively capture the ground-truth transition patterns among items.

One potential limitation of `M`^{2} and the other data-driven basket recommendation methods is that the recommended items may not form realistic baskets. For example, the method may recommend ten brands of milk as a basket to users. However, in practice, users rarely purchase together ten brands in one basket. To mitigate this potential limitation without sacrificing the recommendation performance, we may need to carefully balance the modeling of item complementarities (additional discussions in Section S7) and the other important factors. We leave the investigation of this problem in our future work. In addition to this limitation, another future direction could be to extend `M`^{2} for the cold-start problem. We also leave the investigation of this problem as in our future work.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

This work was supported in part by the National Science Foundation under Grants IIS-1855501, EAR-1520870, SES-1949037, IIS-1827472, and IIS-2133650 and in part by National Library of Medicine under Grants 1R01LM012605-01A1 and R21LM013678-01.

Biographies

Bo Peng is currently working toward the PhD degree with the Computer Science and Engineering Department, The Ohio State University, Columbus, OH. His research interests include machine learning, data mining and their applications in recommender systems and graph mining.

Zhiyun Ren received the PhD degree from the Department of Computer Science, George Mason University, Fairfax, VA, in 2019. Her research interests include machine learning, data mining and their applications in learning analytics, recommender systems and biomedical informatics.

Srinivasan Parthasarathy (Member, IEEE) received the PhD degree from the Department of Computer Science, University of Rochester, Rochester, NY, in 1999. He is currently a professor with the Computer Science and Engineering Department, and the Biomedical Informatics Department, The Ohio State University. His research interests include high performance data analytics, graph analytics and network science, and machine learning and database systems.

Xia Ning (Member, IEEE) received the PhD degree from the Department of Computer Science & Engineering, University of Minnesota, Minneapolis, Minnesota, in 2012. She is currently an associate professor with the Biomedical Informatics Department, and the Computer Science and Engineering Department, The Ohio State University. Her research interests include data mining, machine learning and artificial intelligence with applications in recommender systems, drug discovery and medical informatics.

References

[1]. Hu H and He X, "Sets2sets: Learning from sequential sets with neural networks," in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 1491–1499.

[2]. Rendle S, Freudenthaler C, and Schmidt-Thieme L, "Factorizing personalized Markov chains for next-basket recommendation," in Proc. 19th Int. Conf. World Wide Web, 2010, pp. 811–820.

[3]. Yu F, Liu Q, Wu S, Wang L, and Tan T, "A dynamic recurrent model for next basket recommendation," in Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2016, pp. 729–732.

[4]. Bai T, Nie J-Y, Zhao WX, Zhu Y, Du P, and Wen J-R, "An attribute-aware neural attentive model for next basket recommendation," in Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2018, pp. 1201–1204.

[5]. Yang J, Xu J, Tong J, Gao S, Guo J, and Wen J, "Pre-training of context-aware item representation for next basket recommendation," 2019, arXiv:1904.12604

[6]. He R and McAuley J, "Fusing similarity models with Markov chains for sparse sequential recommendation," in Proc. 16th IEEE Int. Conf. Data Mining, 2016, pp. 191–200.

[7]. Le D-T, Lauw HW, and Fang Y, "Correlation-sensitive next-basket recommendation," in Proc. 28th Int. Joint Conf. Artif. Intell., 2019, pp. 2808–2814.

[8]. Wan M, Wang D, Liu J, Bennett P, and McAuley J, "Representing and recommending shopping baskets with complementarity, compatibility and loyalty," in Proc. 27th ACM Int. Conf. Inf. Knowl. Manage., 2018, pp. 1133–1142.

[9]. Wang W and Cao L, "Interactive sequential basket recommendation by learning basket couplings and positive/negative feedback," ACM Trans. Inf.Syst, vol. 39, no. 3, pp. 1–26, 2021.

[10]. Donkers T, Loepp B, and Ziegler J, "Sequential user-based recurrent neural network recommendations," in Proc. 8th ACM Conf. Recommender Syst., 2017, pp. 152–160.

[11]. Barkan O and Koenigstein N, "Item2Vec: Neural item embedding for collaborative filtering," in Proc. 26th Int. Workshop Mach. Learn. Signal Process., 2016, pp. 1–6.

[12]. Vasile F, Smirnova E, and Conneau A, "Meta-Prod2Vec: Product embeddings using side-information for recommendation," in Proc. 10th ACM Conf. Recommender Syst., 2016, pp. 225–232.

[13]. Mikolov T, Sutskever I, Chen K, Corrado G, and Dean J, "Distributed representations of words and phrases and their compositionality," in Proc. 26th Int. Conf. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[14]. Tang J and Wang K, "Personalized top-n sequential recommendation via convolutional sequence embedding," in Proc. 11th ACM Int. Conf. Web Search Data Mining, 2018, pp. 565–573.

[15]. Kang W-C and McAuley J, "Self-attentive sequential recommendation," in Proc. 18th IEEE Int. Conf. Data Mining, 2018, pp. 197–206.

[16]. Sun F et al. , “BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 1441–1450.

[17]. Ma C, Kang P, and Liu X, “Hierarchical gating networks for sequential recommendation,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 825–833.

[18]. Peng B, Ren Z, Parthasarathy S, and Ning X, “HAM: Hybrid associations models for sequential recommendation,” IEEE Trans. Knowl. Data Eng., early access, Jan. 06, 2021, doi: 10.1109/TKDE.2021.3049692.

[19]. Wang S, Cao L, Wang Y, Sheng QZ, Orgun MA, and Lian D, “A survey on session-based recommender systems,” ACM Comput. Surv, vol. 54, no. 7, pp. 1–38, 2021.

[20]. Hidasi B, Karatzoglou A, Baltrunas L, and Tikk D, “Session-based recommendations with recurrent neural networks,” 2015, arXiv:1511.06939

[21]. Hidasi B and Karatzoglou A, “Recurrent neural networks with top-k gains for session-based recommendations,” in Proc. 27th ACM Int. Conf. Inf. Knowl. Manage., 2018, pp. 843–852.

[22]. Li J, Ren P, Chen Z, Ren Z, Lian T, and Ma J, “Neural attentive session-based recommendation,” in Proc. ACM Int. Conf. Inf. Knowl. Manage., 2017, pp. 1419–1428.

[23]. Liu Q, Zeng Y, Mokhosi R, and Zhang H, “STAMP: Short-term attention/memory priority model for session-based recommendation,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1831–1839.

[24]. Wu S, Tang Y, Zhu Y, Wang L, Xie X, and Tan T, “Session-based recommendation with graph neural networks,” in Proc. AAAI Conf. Artif. Intell., 2019, pp. 346–353.

[25]. Qiu R, Li J, Huang Z, and Yin H, “Rethinking the item order in session-based recommendation with graph neural networks,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 579–588.

[26]. Pal A, Eksombatchai C, Zhou Y, Zhao B, Rosenberg C, and Leskovec J, “PinnerSage: Multi-modal user embedding framework for recommendations at Pinterest,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 2311–2320.

[27]. Koren Y, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2008, pp. 426–434.

[28]. Sarwar B, Karypis G, Konstan J, and Riedl J, “Item-based collaborative filtering recommendation algorithms,” in Proc. 10th Int. Conf. World Wide Web, 2001, pp. 285–295.

[29]. Kameda T and Hastie R, Herd Behavior. Atlanta, GA, USA: American Cancer Society, 2015, pp. 1–14.

[30]. Dean J and Ghemawat S, “MapReduce: Simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[31]. Duchi J, Hazan E, and Singer Y, “Adaptive subgradient methods for online learning and stochastic optimization,” J. Mach. Learn. Res, vol. 12, no. 7, pp. 2121–2159, 2011.

[32]. Guo G, Wang H, Bell D, Bi Y, and Greer K, “KNN model-based approach in classification,” in Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst., 2003, pp. 986–996.

[33]. Cho E, Myers SA, and Leskovec J, “Friendship and mobility: User movement in location-based social networks,” in Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2011, pp. 1082–1090.

[34]. Ying H et al. , “Sequential recommender system based on hierarchical attention networks,” in Proc. 27th Int. Joint Conf. Artif. Intell., 2018, pp. 3926–3932.

[35]. Lika B, Kolomvatsos K, and Hadjiefthymiades S, “Facing the cold start problem in recommender systems,” Expert Syst. Appl, vol. 41, no. 4, pp. 2065–2073, 2014.

[36]. Wiegreffe S and Pinter Y, “Attention is not not explanation,” in Proc. Conf. Empir. Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Proc., 2019, pp. 11–20.

[37]. Dacrema MF, Cremonesi P, and Jannach D, “Are we really making much progress? A worrying analysis of recent neural recommendation approaches,” in Proc. 13th ACM Conf. Recommender Syst., 2019, pp. 101–109.

[38]. van der Maaten L and Hinton G, “Visualizing data using t-SNE,” J. Mach. Learn. Res, vol. 9, no. 86, pp. 2579–2605, 2008.

[39]. Fan Z, Liu Z, Zheng L, Wang S, and Yu PS, "Modeling sequences as distributions with uncertainty for sequential recommendation," in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., 2021, pp. 3019–3023.

[40]. Aggarwal CC, Recommender Systems: The Textbook, 1st ed. Berlin, Germany: Springer, 2016.

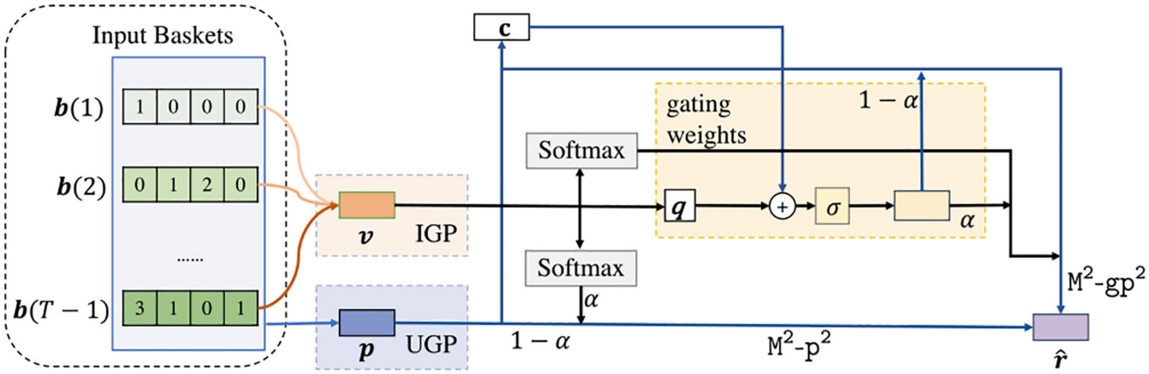

Fig. 1.
M^2-p^2 and M^2-gp^2 model architectures.

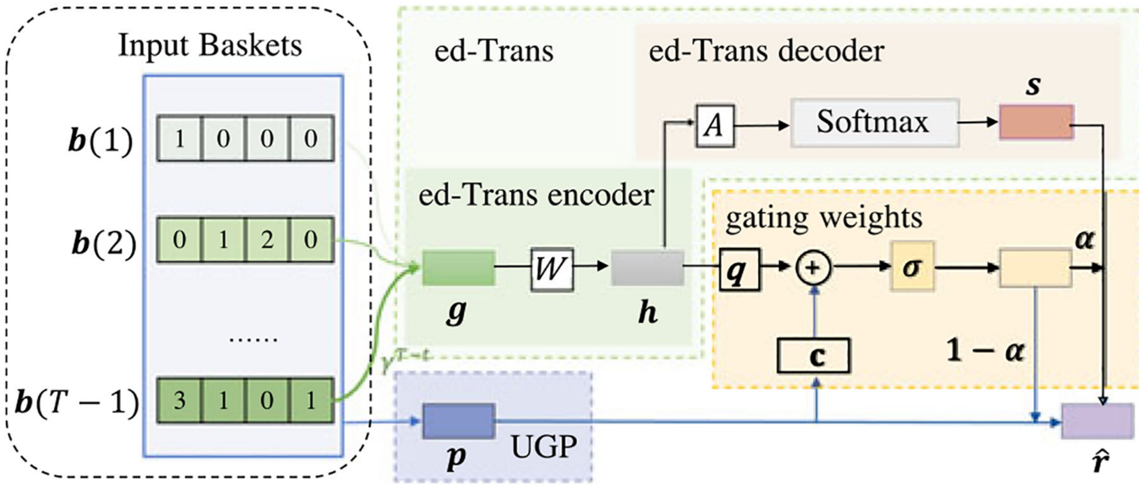

Fig. 2.
M^2-gp^{2}t model architecture.

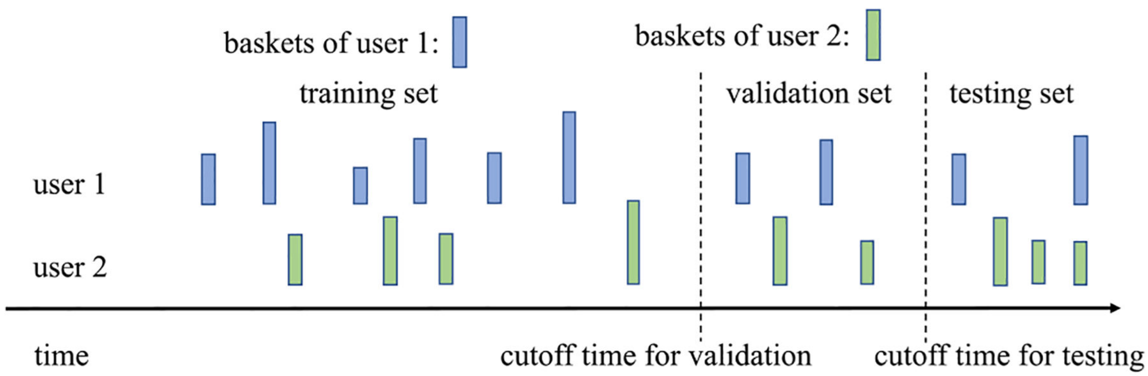

Fig. 3.
M^2-gp^2t datasets splitting protocol.

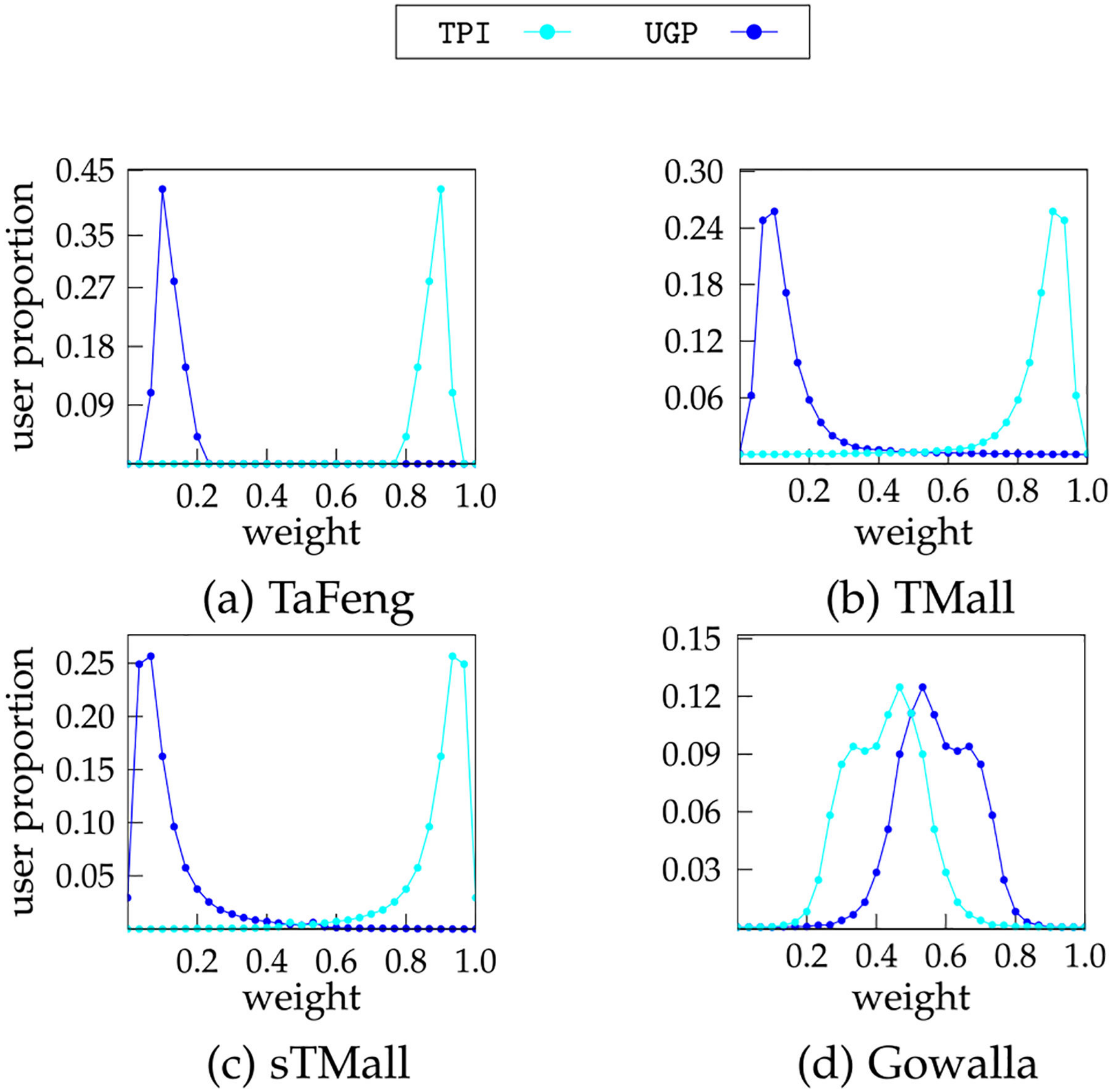

Fig. 4.
Distributions of gating weights from `M²-gp²t`.

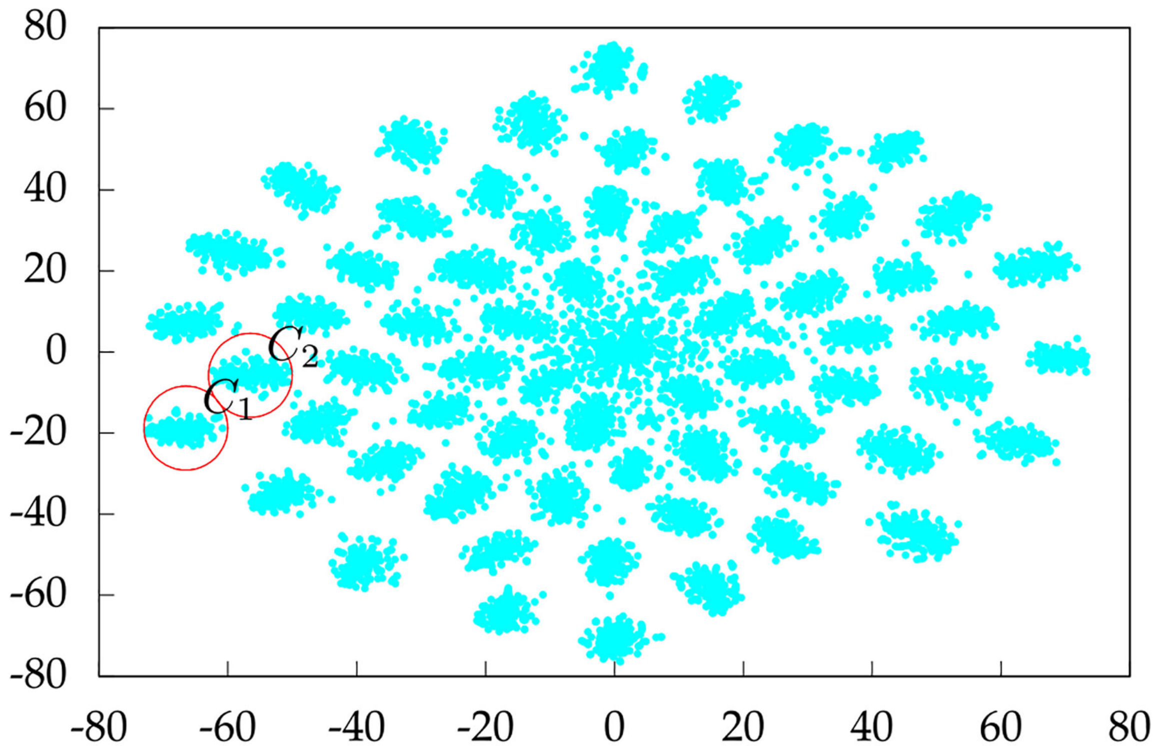

Fig. 5.
Item embeddings from `M`^{2}`-gp`^{2}`t` (TaFeng).

Author Manuscript

Author Manuscript

Author Manuscript

TABLE 1

Notations

notations	meanings
m/n	number of users/items
d	dimension of latent representation of next basket
$B_i/b_i(t)$	the basket sequence/the t-th basket of user i
$T_i/n_i(t)$	the number of baskets in B_i/of items in $b_i(t)$
$\mathbf{r}_i$	the vector representation of the basket $b_i(T_i)$
$\hat{\mathbf{r}}_i$	the recommendation scores over all items for user i

TABLE 2

Dataset Statistics

dataset	#items	#baskets	#users	#items/bskt	#bskt/user
TaFeng	10,829	97,509	16,788	6.72	5.81
TMall	21,812	360,587	28,827	2.41	12.51
sTMall	104,266	2,052,959	214,105	2.01	9.59
Gowalla	26,529	902,505	26,822	1.77	33.65

The columns #items, #baskets, #users, #items/bskt and #bskt/user correspond to the number of items, the number of baskets over all users, the number of users, the average number of items per basket and the average number of baskets per user, respectively.

TABLE 3

Performance Comparison on the Next Basket

	method	recall@*k*			NDCG@*k*		
		***k*=5**	***k*=10**	***k*=20**	***k*=5**	***k*=10**	***k*=20**
TaFeng (7,227)	POP	0.0866	0.0963	0.1151	0.1227	0.1161	0.1203
	POEP	0.0817	0.1153	0.1563	0.1109	0.1127	0.1240
	Dream	0.0839	0.0928	0.1086	0.0694	0.0655	0.0697
	FPMC	0.0568	0.0672	0.0831	0.0691	0.0658	0.0698
	Sets2Sets	0.0822	0.1230	0.1705	0.0952	0.1049	0.1200
	M^2-p^2	0.0908	0.1338	0.1766	0.1192	0.1244	0.1367
	M^2-gp^2	0.0916	0.1344	0.1782	0.1207	0.1257	0.1381
	M^2-gp^2t	†**0.1013**	†**0.1375**	†**0.1936**	†**0.1280**	†**0.1306**	†**0.1469**
	improv	17.0%*	11.8%*	13.5%*	4.3%	12.5%*	18.5%*
TMall (14,051)	POP	0.0802	0.0828	0.0872	0.0777	0.0784	0.0800
	POEP	0.1051	0.1264	0.1524	0.0793	0.0857	0.0927
	Dream	0.0833	0.0868	0.0927	0.0752	0.0765	0.0781
	FPMC	0.0802	0.0809	0.0867	0.0777	0.0778	0.0797
	Sets2Sets	0.1092	0.1360	†0.1653	†0.0979	†0.1071	†0.1154
	M^2-p^2	0.1118	0.1365	0.1584	0.0843	0.0919	0.0977
	M^2-gp^2	0.1123	0.1360	0.1548	0.0846	0.0919	0.0971
	M^2-gp^2t	†**0.1165**	†**0.1395**	**0.1648**	**0.0939**	**0.1010**	**0.1079**
	improv	6.7%*	2.6%*	−0.3%	−4.1%*	−5.7%*	−6.5%*
sTMall (94,337)	POP	0.0859	0.0880	0.0905	0.0834	0.0840	0.0846
	POEP	0.0936	0.1091	0.1187	0.0761	0.0810	0.0836
	Dream	0.0852	0.0873	0.0934	0.0826	0.0833	0.0848
	FPMC	0.0845	0.0869	0.0902	0.0820	0.0828	0.0837
	Sets2Sets	OOM	OOM	OOM	OOM	OOM	OOM
	M^2-p^2	0.0991	0.1203	0.1388	0.0791	0.0857	0.0906
	M^2-gp^2	0.0992	0.1204	0.1393	0.0791	0.0857	0.0907
	M^2-gp^2t	†**0.1114**	†**0.1285**	†**0.1404**	†**0.0948**	†**0.1002**	†**0.1035**
	improv	19.0%*	17.8%*	18.3%*	13.7%*	19.3%*	22.1%*
Gowalla (12,975)	POP	**0.0111**	0.0240	0.0413	0.0064	0.0110	0.0158
	POEP	0.4551	0.5179	0.5649	0.3793	0.4007	0.4136
	Dream	0.0187	0.0307	0.0436	0.0127	0.0169	0.0206
	FPMC	0.0107	0.0255	0.0536	0.0059	0.0111	0.0187
	Sets2Sets	0.3941	0.4745	0.5443	0.3184	0.3462	0.3654
	M^2-p^2	0.4574	0.5213	0.5664	0.3800	0.4019	0.4143
	M^2-gp^2	0.4578	0.5194	0.5689	0.3802	0.4013	0.4148

method	recall@*k*			NDCG@*k*		
	***k*=5**	***k*=10**	***k*=20**	***k*=5**	***k*=10**	***k*=20**
M^2-gp^2t	†**0.4599**	†**0.5232**	†**0.5736**	†**0.3813**	†**0.4030**	†**0.4168**
improv	1.1%*	1.0%*	1.5%*	0.5%*	0.6%*	0.8%*

For each dataset, the best performance among our proposed methods (i.e., M^2-p^2, M^2-gp^2 and M^2-gp^2t) is in **bold**, the best performance among the baseline methods is underlined, and the overall best performance is indicated by a dagger (i.e., †). The row "improv" presents the percentage improvement of the best performing methods among M^2-p^2, M^2-gp^2 and M^2-gp^2t (**bold**) over the best performing baseline methods (underlined) in each column. The numbers in the parentheses after the datasets represent the number of testing users in the datasets. The "OOM" represents the out of memory issue. The * indicates that the improvement is statistically significant at 95 percent confidence level.

TABLE 4

Performance on the Second Next Basket

	method	recall@*k*			NDCG@*k*		
		k=5	*k*=10	*k*=20	*k*=5	*k*=10	*k*=20
TaFeng (2,801)	POP	0.1024	0.1352	0.1475	†0.1356	†0.1392	0.1422
	POEP	0.0920	0.1313	0.1787	0.1056	0.1138	0.1293
	Dream	0.0965	0.1054	0.1168	0.0629	0.0619	0.0651
	FPMC	0.0461	0.0618	0.0805	0.0476	0.0500	0.0558
	Sets2Sets	0.0734	0.1236	0.1882	0.0670	0.0856	0.1086
	M^2-p^2	0.0893	0.1367	0.1927	0.1053	0.1161	0.1345
	M^2-gp^2	†**0.1113**	†**0.1549**	0.2036	**0.1222**	**0.1316**	†**1482**
	M^2-gp^2t	†**0.1113**	0.1517	†0.2062	0.1198	0.1280	0.1461
	improv	8.7%	14.6%*	9.6%*	−9.9%*	−5.5%	4.2%
TMall (5,109)	POP	0.0855	0.0872	0.0892	0.0827	0.0837	0.0844
	POEP	0.1253	0.1556	0.1904	0.0959	0.1052	0.1144
	Dream	0.0907	0.0940	0.0979	0.0844	0.0857	0.0868
	FPMC	0.0860	0.0875	0.0915	0.0831	0.0837	0.0849
	Sets2Sets	0.1345	0.1628	0.1972	†0.1175	†0.1275	†0.1373
	M^2-p^2	0.1344	†**0.1657**	0.1940	0.1019	0.1117	0.1192
	M^2-gp^2	0.1347	0.1645	†**0.2018**	0.1022	0.1112	0.1211
	M^2-gp^2t	†**0.1374**	0.1656	0.1999	**0.1096**	**0.1183**	**0.1276**
	improv	2.2%	1.8%	2.3%*	−6.7%*	−7.2%*	−7.1%*
sTMall (29,741)	POP	0.0835	0.0870	0.0912	0.0820	0.0832	0.0843
	POEP	0.1132	0.1398	0.1563	0.0885	0.0968	0.1011
	Dream	0.0866	0.0893	0.0946	0.0833	0.0842	0.0856
	FPMC	0.0853	0.0874	0.0911	0.0828	0.0835	0.0844
	Sets2Sets	OOM	OOM	OOM	OOM	OOM	OOM
	M^2-p^2	0.1205	0.1482	0.1698	0.0926	0.1012	0.1069
	M^2-gp^2	0.1203	0.1482	0.1699	0.0925	0.1012	0.1069
	M^2-gp^2t	†**0.1258**	†**0.1528**	†**0.1718**	†**0.1023**	†**0.1109**	†**0.1159**
	improv	11.1%*	9.3%*	9.9%*	15.6%*	14.6%*	14.6%*
Gowalla (10,032)	POP	0.0124	0.0228	0.0399	0.0072	0.0110	0.0158
	POEP	0.4765	0.5413	0.5872	0.3920	0.4142	0.4271
	Dream	0.0200	0.0340	0.0507	0.0134	0.0182	0.0228
	FPMC	0.0059	0.0158	0.0329	0.0033	0.0067	0.0112
	Sets2Sets	0.3915	0.4804	0.5565	0.3128	0.3436	0.3646
	M^2-p^2	0.4764	0.5426	0.5894	0.3916	0.4145	0.4275
	M^2-gp^2	0.4767	0.5439	0.5904	0.3921	0.4152	0.4281

method	recall@*k*			NDCG@*k*		
	***k*=5**	***k*=10**	***k*=20**	***k*=5**	***k*=10**	***k*=20**
M²-gp²t	†**0.4787**	†**0.5456**	†**0.5979**	†**0.3932**	†**0.4163**	†**0.4307**
improv	0.5%	0.8%*	1.8%*	0.3%	0.5%	0.8%*

The columns in this table have the same meanings as those in Table 3.

TABLE 5

Performance on the Third Next Basket

	method	recall@*k*			NDCG@*k*		
		***k*=5**	***k*=10**	***k*=20**	***k*=5**	***k*=10**	***k*=20**
TaFeng (1,099)	POP	0.0725	0.1255	0.1519	0.0924	0.1063	0.1142
	POEP	0.1037	0.1415	0.1907	0.1114	0.1207	0.1360
	Dream	0.0632	0.0763	0.0866	0.0552	0.0569	0.0601
	FPMC	0.0420	0.0593	0.0788	0.0451	0.0499	0.0562
	Sets2Sets	0.0732	0.1158	0.1791	0.0650	0.0802	0.1025
	M^2-p^2	0.1039	0.1482	0.1906	0.1109	0.1231	0.1367
	M^2-gp^2	†**0.1162**	†**0.1547**	†**0.2057**	†**0.1205**	†**0.1297**	†**0.1461**
	M^2-gp^2t	0.1141	0.1525	0.1969	0.1162	0.1273	0.1421
	improv	12.1%*	9.3%*	7.9%*	8.2%*	7.5%*	7.4%*
TMall (1,461)	POP	0.0727	0.0741	0.0759	0.0675	0.0678	0.0682
	POEP	0.1522	0.1925	0.2308	0.1096	0.1209	0.1306
	Dream	0.0717	0.0744	0.0799	0.0655	0.0663	0.0677
	FPMC	0.0736	0.0756	0.0791	0.0682	0.0684	0.0696
	Sets2Sets	0.1512	0.1898	0.2368	†0.1256	†0.1387	†0.1517
	M^2-p^2	0.1568	0.1942	0.2348	0.1135	0.1247	0.1348
	M^2-gp^2	0.1586	†**0.1965**	0.2320	0.1150	**0.1260**	0.1349
	M^2-gp^2t	†**0.1603**	0.1909	†**0.2390**	**0.1152**	0.1238	**0.1358**
	improv	5.3%*	2.1%	0.9%	−8.3%*	−9.2%*	−10.5%*
sTMall (7,561)	POP	0.0802	0.0824	0.0854	0.0781	0.0788	0.0795
	POEP	0.1267	0.1610	0.1872	0.0984	0.1087	0.1155
	Dream	0.0838	0.0864	0.0903	0.0800	0.0808	0.0819
	FPMC	0.0824	0.0846	0.0884	0.0792	0.0800	0.0808
	Sets2Sets	OOM	OOM	OOM	OOM	OOM	OOM
	M^2-p^2	0.1348	†**0.1705**	†**0.1961**	0.1029	0.1139	0.1205
	M^2-gp^2	0.1352	0.1703	0.1960	0.1030	0.1137	0.1204
	M^2-gp^2t	†**0.1373**	0.1696	0.1954	†**0.1088**	†**0.1187**	†**0.1254**
	improv	8.4%*	5.9%*	4.8%*	10.6%*	9.2%*	8.6%*
Gowalla (7,985)	POP	0.0108	0.0233	0.0402	0.0062	0.0107	0.0155
	POEP	0.5092	0.5751	0.6251	0.4282	0.4509	0.4649
	Dream	0.0187	0.0295	0.0442	0.0127	0.0166	0.0208
	FPMC	0.0179	0.0404	0.0789	0.0094	0.0172	0.0274
	Sets2Sets	0.4367	0.5230	0.5981	0.3496	0.3799	0.4009
	M^2-p^2	0.5137	0.5779	0.6282	0.4299	0.4518	0.4657
	M^2-gp^2	0.5133	†0.5802	0.6268	0.4296	0.4525	0.4654

method	recall@*k*			NDCG@*k*		
	***k*=5**	***k*=10**	***k*=20**	***k*=5**	***k*=10**	***k*=20**
M^2-gp^2t	†**0.5154**	†**0.5802**	†**0.6321**	†**0.4309**	†**0.4531**	†**0.4675**
improv	1.2%*	0.9%*	1.1%*	0.6%*	0.5%*	0.6%*

The columns in this table have the same meanings as those in Table 3.

TABLE 6

Ablation Study on the Next Basket

	method	recall@*k*			NDCG@*k*		
		***k*=5**	***k*=10**	***k*=20**	***k*=5**	***k*=10**	***k*=20**
TaFeng	UGP	0.0817	0.1153	0.1563	0.1109	0.1127	0.1240
	TPI	0.0508	0.0774	0.1129	0.0660	0.0701	0.0807
	M^2-gp^{2}t	**0.1013**	**0.1375**	**0.1936**	**0.1280**	**0.1306**	**0.1469**
TMall	UGP	0.1051	0.1264	0.1524	0.0793	0.0857	0.0927
	TPI	0.0947	0.1045	0.1162	0.0851	0.0880	0.0915
	M^2-gp^{2}t	**0.1165**	**0.1395**	**0.1648**	**0.0939**	**0.1010**	**0.1079**
sTMall	UGP	0.0936	0.1091	0.1187	0.0761	0.0810	0.0836
	TPI	0.0928	0.0983	0.1052	0.0856	0.0873	0.0892
	M^2-gp^{2}t	**0.1114**	**0.1285**	**0.1404**	**0.0948**	**0.1002**	**0.1035**
Gowalla	UGP	0.4551	0.5179	0.5649	0.3793	0.4007	0.4136
	TPI	0.3105	0.3342	0.3567	0.2778	0.2856	0.2917
	M^2-gp^{2}t	**0.4599**	**0.5232**	**0.5736**	**0.3813**	**0.4030**	**0.4168**

M^2-gp^{2}t is identical to UGP+TPI. The best and second best performance in each dataset is in **bold** and underlined, respectivley.